Why the standard methods, 5GL, common platforms and reusable components are the four pillars of the new computational paradigm Programming without programmers

Ivan Stanev, Maria Koleva

***Abstract*—**Two important problems concerning software engineering are identified: low efficiency of the software development process, and lack of programmers. Three key components with considerable impact on the software development process, namely the software engineering method, the problem specification language, and the platform used, are identified. Four state of the art reviews are prepared including software engineering methods, problem specification languages, cloud computing platforms, and related works. This research problem is solved by the realization of the Knowledge Based Automated Software Engineering (KBASE) method for generation of enterprise information systems, based on 5GL specification languages, common platform for automated programming and a set of intelligent components for generation of new applications from well-defined system and business reusable components in three groups – domain independent, domain dependent and problem solving. KBASE concept, KBASE technological process, KBASE specification language, and KBASE platform are described. Key results achieved in the related works are detailed. The efficiency of the new computational paradigm is presented.

***Keywords*—**architecture-based, knowledge based automated software engineering, ontology, business architecture; software architecture; infrastructure architecture.

I. Introduction

The overall IT spending in 2018 for enterprise information systems (data center systems, devices, enterprise software, and IT services) is projected to 2.2 trillion USD [14]. The enterprise software and IT services markets form 60% of the overall IT spending worldwide. Its average annual growth for 2017-2022 is forecasted at 13.8% – the highest expected growth for this period. At the same time, the development process of Enterprise Information Systems (EIS) is still less effective and more resource consuming than the software industry and their clients would require.

I. Stanev is with the Computer Informatics Department of University of Sofia "St. Kliment Ohridski", Sofia 1000, Bulgaria (phone: +359 2 8161 508, e-mail: instanev@fmi.uni-sofia.bg).

M. Koleva is with the Computer Informatics Department of University of Sofia "St. Kliment Ohridski", Sofia 1000, Bulgaria, and the Informatics and Information Technologies Department of University of Ruse "Angel Kanchev", Ruse 7000, Bulgaria (e-mail: mkoleva@fmi.uni-sofia.bg).

The demand for software developers is rapidly increasing. According to the U.S. Bureau of Labor Statistics [31], software developer jobs are expected to grow much faster than the average - with 24% from 2016 to 2026. The main reasons for this tendency are due to: (1) the increasing demand of software products; (2) the insufficient efficiency of widely used software development methods and tools.

This paper presents a solution to these problems by introducing the new computational paradigm Programming without programmers. The new paradigm is realized through the Knowledge Based Automated Software Engineering (KBASE, [7]) method.

II. State of the art

The efficiency of the software development process (SDP) is a function of its three main components: the software engineering method, the problem specification language, and the platform used.

Four ***state of the art reviews*** are prepared including: Software Engineering (SE) methods, problem specification languages, cloud computing platforms, and KBASE related works.

The **Software engineering** (SE) **methods** can be split in three groups: (1) methods covering only the management aspects of the SDP, called "shell" methods such as: PRINCE2 (Projects IN Controlled Environments [2]); TEMPO (TAXUD Electronic Management of Projects Online); PMBOK (Project Management Body of Knowledge [27]); (2) methods partially covering the management and technological aspects of rapid SDPs (the Agile family of methods [23]); (3) methods covering to a great extent the management and technological aspects of SDPs such as Rational Unified Process (RUP [24]).

The main strength of the ***Shell group*** is the well-defined project management process. The key weakness of these methods is the lack of standardized technological process. Therefore, these methods are neither efficient nor appropriate for the SDP automation.

The key strength of the ***Agile group*** is the method simplicity and project management efficiency. The important weakness of these methods is the lack of consistent design. This reduces

to minimum the software reusability and make the methods inappropriate for developing Enterprise Information Systems (EIS).

The important strength of the ***RUP group*** is the well-defined IT technological process. The key weaknesses of these methods are their complexity and the expensive development process. The complexity of the methods makes the SDP and testing too heavy and inefficient.

To overcome these problems a new SE method shall be defined combining the Agile and RUP strengths. This method shall improve the simplicity and efficiency of the software development process by introducing reusable components and automation of the process.

A number of **problem specification languages**, including the most disseminated imperative languages Java and C#, the declarative languages ML and Prolog, the Generative Modelling Languages UML and BPMN, the fifth generation modelling language Net, and a Natural Language Processor are analyzed in [6].

The main strengths of the ***Imperative Languages*** (IL) ***group*** (**Java**, **C#**) concern the fast runtime performance, the availability of rich sets of supporting libraries and third party components, the good platform portability. The key weaknesses of this language group refer to the lack of incorporated problem solver (i.e. specification of the problem being solved and specification of the algorithm for problem solving), and the huge efforts for design, implementation, and testing by highly qualified IT specialists. So, nevertheless that these languages are of great industrial importance, they are neither efficient nor appropriate for SDP automation.

The key strengths of the ***Declarative Languages*** (DL) ***group*** (**ML**, **Prolog**) concern the incorporated problem solver (i.e. only specification of the problem being solved is required, thus DL are suitable for automated programming), and the small design, implementation, and testing effort by IT specialists having relatively low qualification. The important weaknesses of this language group refer to the slow runtime performance (e.g. the machine arithmetic), and the lack of tools for design of complex problems. The DLs are therefore inappropriate for EIS specification and development.

The important strengths of the ***Generative Modelling Languages*** (GML) ***group*** (**BPMN**, **UML**) are the well-standardized modelling capabilities, and the small design effort of domain area experts and IT specialists with relatively low qualification. Their key weakness is the considerable gap between the design and implementation models. Hence, GML are not appropriate for realization of Enterprise IS, while the SE methods based on GML are not efficient for the whole SE life cycle.

The important strength of the ***Fifth Generation Languages*** (5GL) ***group*** (**Net**) concerns the availability of incorporated tools for intelligent problem solving. Their key weakness refers to the insufficient set of supporting libraries and third party components. The 5GL are therefore not appropriate for realization of Enterprise IS, and the SE methods based on the 5GL are inefficient.

The important strength of the ***Natural Language Processing*** (NLP) ***group*** (**Bulgarian Language Processor**) is the effortless use of NL human-machine interfaces for all native speakers of this language. The key weaknesses refer to the low reusability of the specified components, and the lack of unified techniques for description of NL semantics. The use of NLP alone as a tool for design, implementation, and testing makes the processes too heavy and inefficient.

In order to overcome these problems a specification language shall be created as a combination of the strengths of IL, DL, GML, 5GL and NLP. This language shall improve the simplicity and efficiency of problem specification. For that purpose, the language shall be a GML and 5GL standardized modelling language covering the whole SE life cycle. The language shall be equipped with NLP, it shall include a problem solver and shall have the possibility for automated use of wide range of supporting libraries and third party components.

The industrial **cloud** [25] **platforms** [5] Amazon **AWS**, Microsoft **Azure**, Google **App Engine**, VMWare **vCloud**, IBM **Bluemix**, HP **Helion**, and Oracle **OCPaaS** are analyzed in [11] as referred to.

The important strengths of the platforms concern the high-level integration of hardware and software tools in the cloud, and the good quality of tools for control of distributed computation in the cloud. Their key weakness refers to the insufficient use of tools for intelligent self-organization, as well as the state engine and multi-agent control. This weakness makes the existing platforms too expensive and inefficient for realization of complex ISs.

To overcome these difficulties, the existing platforms are extended with new tools for specification of the state engine and multi-agent architectures, and the library of intelligent components for platform control and self-organization.

Some important **related works** of the KBASE team demonstrating the SDP efficiency improvement capabilities of the KBASE concept refer to (1) the Module for Automated Programming of Robots (**MAPR** [6]) which generates programs for Robot Control based on a natural language specification; (2) the Intelligent Product Manual (**IPM** [7]) which is a software system generating Product Manuals for manufacturing products; (3) the Built-in-Test Adaptive Document Display (**BIT** [26]) which realizes the automation of Contract tests process, during the integration of a Commercial-off-the-shelf component with hosting system and the automation of Quality of Service tests process during real-time operation; (4) the Information Objects Manager (**IO.Man** [7]) which is a tool generating software components for document management through information objects structure and behavior formal specification; (5) Bulgarian e-Customs (**BeC** [12]), (6) Bulgarian e-Health (**BeH** [10]) and (7) Bulgarian e-Government (**BeG** [9]), which represent three national EIS.

Four **research questions** (RQs) are identified analyzing the

related works: **RQ1** How to generate machine code directly from Business Architecture specification? **RQ2** How to transfer the key responsibility for the software development process from IT experts to DA experts? **RQ3** How to replace handmade programming with intelligent self-organizing components in the cloud platform? **RQ4** How to improve component reusability by introducing domain dependent, domain independent and problem solving phases in software development process?

This **research problem** is solved by the realization of the standardized method KBASE for generation of enterprise ISs, based on 5GL specification languages, common platform for automated programming (CPAP) and a set of intelligent components for generation of new applications from well-defined system and business reusable components split in three groups – domain independent, domain dependent and problem solving.

III. Research Methodology

The ***research methodology*** actions are grouped in the following 6 categories: (1) ***SE state-of-the-art reviews***. State-of-the-art reviews, including reviews of SE Methods, SE computational paradigms, SE Specification techniques, and SE Platforms are performed. The KBASE objectives are defined after analysis of the results. (2) ***KBASE case studies analysis***. Several different types of information systems are selected for analysis in accordance with the following criteria: (a) IS must be in the context of KBASE and relate to different parts of the KBASE concept; (b) IS must have well defined intelligent functions; (c) IS should use GML, 5GL, or NL specification language; (d) IS should have a full set of available detailed deliverables. The selected systems are the Rule Based Information System Umcho1 [28], the Embedded information system OSCAR [13], the Manufacturing Information System IPM [3], the Distributed Information System BeG [4], the Self-organized Information System IPM-DD [26]. **(3)** ***KBASE Method creation***. Definition of KBASE IT perspective including description of the KBASE concept, as well as method phases, objects, and components. KBASE technological process is defined to cover roles, disciplines, and artefacts. **(4)** ***KBASE Specification Language creation***. Description of KBASE full set of Specification Techniques. **(5)** ***KBASE Platform creation***. Detailed description of CPAP structure and components. **(6)** ***KBASE results***. Description of Case Studies results based on the KBASE method, SL, and CPAP.

IV. The Method Concept

The main KBASE objective is the creation of a SE method and a tool for IS generation, which significantly improve the efficiency of the software development process through: (1) reduction of SE method steps by generating machine code directly from the business architecture, excluding the software architecture design steps, (2) creation of specification language for programming with limited involvement of programmers, and (3) inclusion of intelligent capabilities in the proposed platform.

The ***KBASE concept*** (Fig. 1) is based on three categories of objects: Business Models (BM), Business Model Templates (BMT), and Virtual Reality Model (VRM). Business models are used for specification of the required IS. Business Model Templates are predefined reusable parts of BM, which save design time and shorten the specification process. Virtual reality model is a predefined model of the Domain Area describing BM structure, behavior, and constraints. BM is verified against VRM for completeness, precision, and readiness for IS generation.

The **VRM** is a subject of ***interdisciplinary research***. The purpose of this research is to find the intersection of the **three disciplines** – **Software Engineering**, **Economics**, and **Linguistics**. The created VRM covers the following requirements: (1) VRM is common for all three disciplines; (2) VRM is established of three views (SE, Economics, and Linguistics), which are integrated in a common structure without collisions; (3) VRM is described using specification techniques, which are natural for each of the three different groups of experts, and easy to understand and use by the other two groups; (4) the VRM describes the structure, behavior, and constraints of the virtual images of the real world objects used in the three disciplines; (5) the VRM contains the minimum information required by the three Object Verifiers to confirm the BMs readiness for machine code generation.

The term "**generation**" means generation of objects, components or instruments for integration and / or parametrization of pre-defined objects and components. Nevertheless that the whole IS could be generated, the typical generation process within the scope of the proposed project refers to the generation of a machine code for integration and / or parametrization of pre-defined objects and components.

The generated IS is specified by a **Business model** that includes three types of components: Business control structures, Business objects, and Business architecture. All Business model components are built by composite objects, primitive objects and data objects. Composite objects are the control structures, namely, the business processes, the state engines, and the multi-agent systems. The primitive objects are components, services, interfaces, messages, business rules and third party components. The data objects are parameters of the business models, parameters of the generated IS, and/or data of the solved problems.

BM, BMT and VRM objects are stored in two **Knowledge Bases** – one for composite objects and one for primitive objects, and in one Data Base – for data objects. All three repositories are used in synergy during the domain dependent, domain independent and problem solving phases of the process.

BM is specified in the terms of a **Specification Language** (SL) which has the characteristics of the fifth generation declarative modelling languages. SL enables incomplete and imprecise specification based on the used standard modelling

languages (BPMN, UML, CMMN, DMN), the fifth generation language Net, and the restricted natural language. SL specifications include definitions of composite objects, descriptors of primitive objects, and descriptors of data objects. SL covers the whole SE method life cycle.

BM specifications **formal validation** is realized by three processors (for composite, primitive, and data objects). The **semantic verification** for completeness and precision of BM specifications is carried out by three **intelligent verifiers**. The verifiers compare the BM specifications with the pre-defined domain area VRM. If any incompleteness and / or imprecision is found, the verifiers remove them or request additional information from the experts in order to remove them.

Once completeness and precision necessary to solve the problem are reached, the business specifications are passed on to a **product generator**. The verified BM specifications, and the repositories containing the predefined Primitive objects and Data objects, are inputs to the Product Generator. The generated IS is the output of the Product Generator. The role of the Product Generator is to generate the machine code for BM Composite objects, and to map the descriptors of primitive and data object to their machine code stored in the relevant repositories.

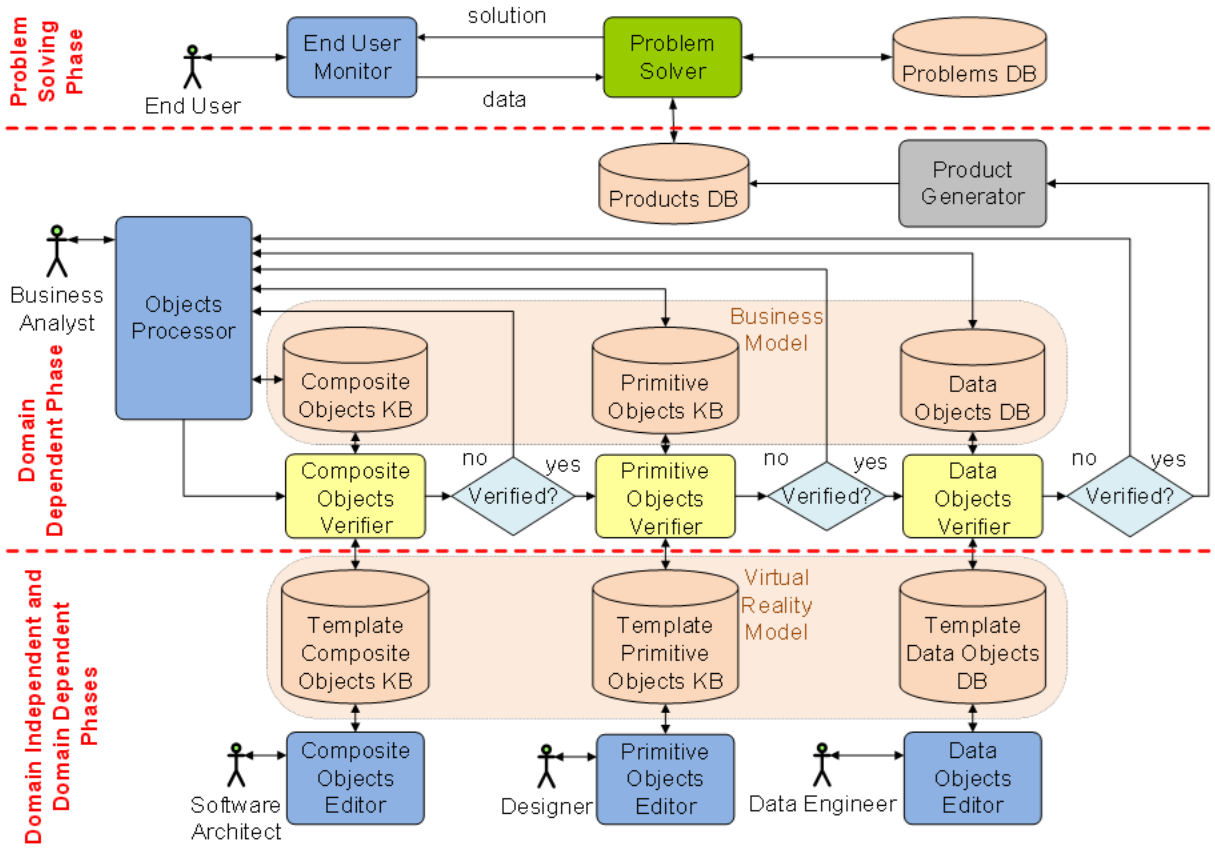

Fig. 1. KBASE Concept

The software development process is divided into three main phases, each with IT Perspective and Domain Area Perspective: (1) The **Domain Independent Phase** involves the software architect, the designer and the data engineer. They prepare the VRM and BMT complex objects, primitive objects, and data objects, which are suitable for various domain areas. Usually, these objects are developed once; they are re-usable and are subject to minor changes only in exceptional cases; (2) The **Domain Dependent Phase** involves the software architect, the designer and the data engineer. They prepare the VRM and BMT composite objects, primitive objects, and data objects suitable for the design domain area. This phase also involves the business analyst who designs the BM on demand by the end users. If necessary, the business analyst requires relevant modifications of VRM and BMT to be made by the software architect, the designer and the data engineer. The product is generated; (3) **Problem solving phase** involves the end user who uses the generated product for solving a specific problem based on the input data that he has provided.

V. KBASE TECHNOLOGICAL PROCESS

The Technological Process is intended to standardize the KBASE development process. The KBASE development process is detailed in [8] and defines: (1) the technological ***roles*** responsible for the work performed within each discipline; (2) the ***disciplines*** presenting the actions performed throughout the entire lifecycle of the KBASE product including the specification techniques used (cf. section II); and (3) the resulting ***artefacts*** including the minimum full set of

artefacts required for quality development and maintenance of KBASE products.

The KBASE technological process combines the Agile and RUP strengths (cf. section II) in order to improve the efficiency of the software development. For the purpose of the KBASE method, the following simplifications/ extensions are proposed: **(1)** the ***pre-project*** is performed by implementing the disciplines and actions to a limited extent; **(2)** the ***technological roles*** include mainly the groups of roles prescribed in the RUP. The Knowledge Engineer role is added to cover activities related to automated programming and knowledge processing; **(3)** the ***artefacts set*** contains the most important and most commonly used RUP artefacts including the minimum full set of artefacts required for quality development and maintenance of KBASE products; **(4)** the ***disciplines*** include RUP disciplines with small modifications, namely Project Management, Requirements, Infrastructure, Analysis, Testing, Design, Generation and Implementation, Exploitation; **(5)** the RUP ***discipline Business Modeling*** is removed since low productive, and highly resource consuming; **(6)** the RUP discipline Analysis and Design is separated in two KBASE disciplines – first ***Analysis***, and second ***Design***, in order to reduce the gap between domain area experts and analysts on the one hand and between analysts and designers on the other hand; **(7)** the RUP discipline Implementation is replaced by ***Generation & Implementation***. This discipline covers the process of automated generation of KBASE products from incomplete and imprecise specifications.; **(8)** the RUP horizontal disciplines Configuration and Change management, Project management, and Environment discipline process aspects are combined in one ***Management discipline*** to optimize project realization. The "Management" discipline represents horizontal activities related to planning, specification and realization; **(9)** the KBASE Requirements discipline uses the ***Contextual design*** models which improve the quality of primary information collection; **(10)** the KBASE Analysis discipline uses the ***process-based***, ***object-based*** and ***event-based*** specification techniques instead of the RUP Use Case Specification in order to reduce the resources for UML model completion; **(11)** the KBASE Design discipline uses ***ontologies*** for automation of the development process; **(12)** the KBASE Generation & Implementation discipline uses ***automated support of models and documents***; **(13)** the Test discipline actions uses ***structured Use Case specifications*** for better compliance between use cases and test cases, improvement of the quality and efficiency of the test process; **(14)** the Test discipline actions uses the ***Built-in-Test method*** ([26]) for automated testing; **(15)** the actions from the RUP Deployment discipline are incorporated in the KBASE ***Exploitation discipline***; **(16)** the infrastructure aspects of the RUP Environment discipline are incorporated in the KBASE ***Infrastructure discipline***; **(17)** the "***infrastructure diagram***" term is used instead of the UML standard "deployment diagram" to better convey the purpose with no changes in the notation; **(18)** the "***Infrastructure model***" term is used instead of the standard RUP "Deployment model" to incorporate the development, testing, pre-production, production and management environments.

As a result of KBASE technological process introduction in practice two important changes for the software companies are identified: (1) during the first 2-3 projects performed in a new domain area the productivity of the company is decreased by 10 – 20 %; (2) for every following project in the same domain area the productivity of the same company is increased by 40 – 70 %.

VI. KBASE Specification Language

The actions of the method presented in section I are implemented using nine Specification Techniques (ST) for both formal and natural language specifications (TABLE I.). They are detailed in [8]. The ST are used to specify the different objects in the different phases of the process. These techniques are linked (TABLE II.) with the steps of the process, the artefacts prepared and the actions taken. The specification techniques are implemented using wholly or partly some of the most common specification standards.

TABLE I. Specification techniques and standards

Technique	Standard
NL based	**NL Combinatorial Dictionary**
context based	**Contextual Design** [16] (all models)
event based	Case Management Model and Notation (**CMMN**, [20])**,** Business Process Model and Notation (**BPMN**, [19]), Unified Modeling Language (**UML**, [22]) - component, deployment, and package diagrams
process based	**BPMN**
message based	**BPMN, UML** - sequence diagram
service based	**UML** - all diagrams
object based	**UML** state engine
rule based	Decision Model and Notation (**DMN**, [21])
ontology based	Web Ontology Language (**OWL**, [32])

The ST are used for KBASE specification preparation and processing including syntactic and semantic, as well as interoperability aspects. An automated programming technique based on reuse of objects and components from KBASE repositories is applied during the three Specification Phases (cf. section IV).

The automated programming process is presented in TABLE II. below. The disciplines names are given in the "Disc." column. The work performed within each discipline is described in the "Discipline action" column. The specification technique/s used for the relevant action is indicated in the last column "Sp.technique" and it is further described in [8]. The resulting artefact is listed in the "Art." column.

TABLE II. KBASE lifecycle disciplines

Disc.	Art.	Discipline action	Sp.technique
Mng	PMP	project management and realization	SPEM
Mng	PMP	configuration and change management and realization	SPEM
Mng	QMP	quality management and realization	SPEM

Disc.	Art.	Discipline action	Sp.technique
Req	RM	gather and analyze relevant legal base	context based, NL based, ontology based
Req	RM	build organization hierarchy	object based
Req	RM	define functional areas	context based
Req	RM	build roles hierarchy for each functional area	object based
Req	RM	develop software architecture concept	event based
Req	RM	gather functional requirements	context based, NL based, ontology based
Req	RM	gather non-functional requirements	context based, NL based, ontology based
Req	RM	categorize, classify and prioritize functional requirements	context based
Req	RM	categorize, classify and prioritize non-functional requirements	context based
Inf	InfM	infrastructure management and realization	event based
A	BM	selection of control component (business process / state engine / multi-agent system)	process based / object based / event based
A	BM	control component description	process based / object based / event based
A	BM	high level description of control components tasks (task type, input, output and system objects, interfaces, interactions, messages, reports, security, actors, preconditions, post conditions)	service based, rule based
A	DatM	identify primary data objects - documents, forms, reports, messages	object based, rule based
A	DatM	develop data objects hierarchy	object based
A	BM	select candidate tasks	event based
A	BM	select tasks	event based
A	BM	refine control component with candidate tasks to control component with tasks	event based
A	UCM	define use cases for each task	service based
A	UCM	define input and output data objects for each task and select data objects storage options	object based, rule based
A	UCM	define input and output messages for each task and select messages storage options	object based
T	TM	test management and realization	BIT
D	SA	develop software architecture component diagrams	event based
D	SA	define system architecture components	event based
D	SA	define interfaces of system architecture components	interface based
D	SA	develop GUI navigation tree	object based
D	DesM	prepare class diagrams of the tasks	service based
D	DesM	define attributes and operations of classes	service based
D	DesM	develop sequence diagrams of the tasks	interface based
D	DatM	refine object data model into relational data model	object based
D	DatM	relational data model normalization (if necessary)	object based
D	DesM	develop statechart diagrams of the tasks	interface based
D	DesM	develop statechart diagrams of important data objects	interface based
D	DatM	update Data model	object based, rule based
D	ImpM	prepare package diagrams	event based
D	OM	associate use case, component, class, package and infrastructure diagrams	ontology based
G&I	ImpM	code generation	all
G&I	ImpM	code implementation	all
G&I	ImpM	code integration	all
Exp	all	deployment	all
Exp	all	exploitation	all
Exp	all	enhancement	all
Exp	all	recycling	all

Abbreviations: **A** – Analysis discipline, **BIT** - Built-in-Test ([26]), **BM** – Business model, **D** – Design discipline, **DatM** – Data model, **DesM** – Design model, **Exp** – Exploitation discipline, **G&I** – Generation and Implementation discipline, **ImpM** – Implementation model, **Inf** – Infrastructure discipline, **InfM** – Infrastructure model, **Mng** – Management discipline, **OM** - Ontology model, **QMP** – Quality Management Plan; **PMP** – Project Management Plan, **Req** – Requirements discipline, **RM** – Requirements Model, **SA** – Software Architecture, **SPEM** - Software & Systems Process Engineering Metamodel Specification ([18]); **T** – Test discipline, **TM** – Test model, **UCM** – Use case model.

As result of KBASE specification language introduction in practice two important changes for the software companies are identified: (1) development time is reduced with 40 – 60 % (2) the business analysts are increased twice, and the implementers are decreased more than three times.

VII. KBASE PLATFORM

The two key roles of the KBASE platform are: (1) the single unified control of the generation and performance of all KBASE products; (2) the creation, management and support of a well standardized structure of reusable components.

The single unified control of the generation and performance of all KBASE products is realized by the Common Platform for Automated Programming (CPAP), which is built on the following principles:

(a) **CPAP components** are arranged in: (1) Infrastructure Layer; (2) Cloud Layer; (3) Control Layer; (4) DB Layer; (5) Development Layer; (6) Operational Layer; (7) Knowledge Layer; (8) Management & Planning Layer; (9) Integration Layer. The layers communicate through a standard and semantic Enterprise Service Bus.

(b) **CPAP components** are designed to build: (1) **embedded systems** – for collecting information; (2) **information systems** – for operational information processing; (3) **knowledge based systems** – for automated

processing of large data sets, for generating new components and for monitoring and re-configuration of CPAP; (4) **management and planning systems** – for reports generation, statistical data, trends, plans, and prognosis production.

(c) The Embedded Systems Manager, the Information Systems Manager, and the KBASE Manager in P12 are **CPAP master managers**. The master managers supported by their respective models in P10, and integration definitions in P28, generate all non-master managers and applications.

(d) CPAP **non-master managers** in P12, supported by their respective models in P10, control the runtime performance of their applications.

(e) The **static and dynamic behavior of the master components and systems** is formally described by all P11 master models, and all P28 integration definitions using one or more SL components.

(f) The **static and dynamic behavior of the non-master components and systems** is formally described by all P11 non-master models, using one or more SL instruments.

(g) The **collaboration between CPAP and its external users** (e.g. economic operators) is organized by **Template Information Systems** (TIS) generated by the **TIS Manager**. The TIS Manager is managing the IS Manager, KBASE Manager, and Embedded Systems Manager for the purposes of collecting data from external systems. The TIS Manager manages the generation of the TIS meta-model in P11 and TIS integration definitions in P28 for internal and external use. The TIS Manager subsequently generates TIS in collaboration with the Embedded Systems Manager, Information Systems Manager, and Knowledge Base Manager.

The hierarchical structure created in KBASE CPAP is based on the first CPAP classification [11]) and three widely disseminated IS classifications, namely by Laudon ([15]), Nevo ([30]), and Ralston ([1]).

The main strength of the first CPAP classification is the well-defined system and business components. The key weakness is the structure of the business applications layer.

The main strength of the Laudon classification is the well-classified business components. The key weakness is the coarse grained classification of system components.

The main strength of the Nevo classification is the balanced presentation of hardware, firmware and software. The key weakness is the incompleteness of component types.

The main strength of the Ralston classification is the well-defined set of IS component types. The key weakness is the insufficient detail in identified business components.

The CPAP classification is a combination of the strengths of the above mentioned classifications. The CPAP architecture is composed of nine layers:

L1 Infrastructure Layer organizes and manages the hardware and communication infrastructure processes at physical level (P01 Hardware) and operating system level (P02 Real OS) using virtualization tools (P03 Virtualization).

TABLE III. LAYER 1

Layer	Pack	Comp	Layer/ Package/ Component
L1	**0**	**0**	**Infrastructure Layer**
L1	**P01**	**0**	**Hardware**
L1	**P02**	**0**	**Real OS**
L1	**P03**	**0**	**Virtualization**

L2 Cloud Layer automatically organizes the execution of the requested work in the cloud environment. It is performed under the control of the virtual operating system (P04 Virtual OS) using scaling mechanisms (P05 Cloud instances), and allocating the required physical and virtual resources (P06 Cloud cartridges).

TABLE IV. LAYER 2

Layer	Pack	Comp	Layer/ Package/ Component
L2	**0**	**0**	**Cloud Layer**
L2	**P04**	**0**	**Virtual OS**
L2	**P05**	**0**	**Cloud - instances**
L2	**P06**	**0**	**Cloud - cartridges**

L3 Control Layer manages the execution of user assignments at service level (through P07 Application Servers) and process level (through P08 Control Servers).

TABLE V. LAYER 3

Layer	Pack	Comp	Layer/ Package/ Component
L3	**0**	**0**	**Control Layer**
L3	**P07**	**0**	**Application Servers**
L3	P07	1	KBASE Indexing Server
L3	P07	2	Runtime Server
L3	P07	3	ODB Server
L3	P07	4	Web Server
L3	P07	5	RDB Server
L3	P07	6	Application Server
L3	P07	7	File Server
L3	P07	8	Security Server
L3	P07	9	Directory Server
L3	P07	10	Identity Server
L3	P07	11	PKI Server
L3	P07	12	Document Server
L3	P07	13	Search Engine
L3	P07	14	Test Server
L3	P07	15	Email Server
L3	**P08**	**0**	**Control Servers**
L3	P08	1	KBASE Server
L3	P08	2	Process Server
L3	P08	3	Case Server
L3	P08	4	Interface Server

L4 DB Layer contains data (P09 Data Base), components and services (P10 Repository), and knowledge (P11 Knowledge Base) necessary for generating the required software products by using BPMN and/ or UML specifications, graphical interfaces, natural language, etc. The models included in P11 are subject to various automated verifications such as verification and modification based on predefined standardized knowledge bases [7], business process model quality assessment, and evaluation of business process semantic correctness.

TABLE VI. LAYER 4

Layer	Pack	Comp	Layer/ Package/ Component
L4	**0**	**0**	**DB Layer**
L4	**P09**	**0**	**Data Base**
L4	P09	1	RDB
L4	P09	2	Data Warehouse
L4	**P10**	**0**	**Repository**
L4	P10	1	Components
L4	P10	2	Services
L4	**P11**	**0**	**Knowledge Base**
L4	P11	1	TIS Model
L4	P11	2	KBASE Model
L4	P11	3	IS Model
L4	P11	4	Embedded Systems Model
L4	P11	5	Case Model
L4	P11	6	Process Model
L4	P11	7	Form Model
L4	P11	8	Report Model
L4	P11	9	Service Model
L4	P11	10	Interface Model
L4	P11	11	Object DB Model
L4	P11	12	Relational DB Model
L4	P11	13	Security Model
L4	P11	14	Test Model
L4	P11	15	Document Model
L4	P11	16	GIS Model

L5 Development Layer contains a rich set of ready-made tools that are provided on demand by the user to assemble their software product. Components in L5 are organized in two packages. The first package consists of components for design management (P12 Design management) responsible for managing the process of creating new software products in the context of CPAP and updating them on demand (possibly in real time) [6], [7]. The second package consists of components for runtime management (P13 Runtime management) responsible for managing the process of runtime performance of software products in the context of CPAP.

TABLE VII. LAYER 5

Layer	Pack	Comp	Layer/ Package/ Component
L5	**0**	**0**	**Development Layer**
L5	**P12**	**0**	**Design management**
L5	P12	1	KBASE Manager
L5	P12	2	TIS Manager
L5	P12	3	IS Manager
L5	P12	4	Embedded Systems Manager
L5	P12	5	Case Manager
L5	P12	6	Process Manager
L5	P12	7	Service Manager
L5	P12	8	Object Manager
L5	P12	9	Interface Manager
L5	P12	10	Test Manager
L5	P12	11	GUID Manager
L5	P12	12	GIS Manager
L5	P12	13	Security Manager
L5	P12	14	Report Manager
L5	P12	15	Admin Manager
L5	P12	16	DB Manager
L5	P12	17	Content Manager
L5	P12	18	History Manager
L5	P12	19	Rule Manager
L5	P12	20	Code list Manager
L5	**P13**	**0**	**Runtime Management**
L5	P13	1	Automation Manager
L5	P13	2	Cloud Services Manager
L5	P13	3	Runtime Manager
L5	P13	4	Business Activity Manager

L6 Operational Layer consists of components (typically reusable) for management of the work at business applications operational level. The components here are split in five packages: P14 Organization Management, P15 Document Management, P16 Activity Management, P17 Collaboration Management, and P18 User Support.

TABLE VIII. LAYER 6

Layer	Pack	Comp	Layer/ Package/ Component
L6	**0**	**0**	**Operational Layer**
L6	**P14**	**0**	**Organization Management**
L6	P14	1	Organization Manager
L6	P14	2	Mandate Manager
L6	P14	3	Party Manager
L6	P14	4	Roles Manager
L6	P14	5	Policy Manager
L6	**P15**	**0**	**Document Management**
L6	P15	1	Document Manager
L6	P15	2	Register Manager
L6	**P16**	**0**	**Activity Management**
L6	P16	1	Legal Base Manager
L6	P16	2	Financial Manager
L6	P16	3	Payment Manager
L6	P16	4	Point of Sales systems (POS)
L6	P16	5	HR Manager
L6	P16	6	Customer Relationship Manager
L6	P16	7	Supply chain Manager
L6	P16	8	Marketing Manager
L6	P16	9	Procurement Manager
L6	P16	10	Sales Manager
L6	P16	11	Enterprise Resource Planning Manager
L6	P16	12	Assets Manager
L6	P16	13	Purchase Order Manager
L6	P16	14	Machine Control Manager
L6	P16	15	Plant Scheduling Manager
L6	P16	16	Quality Control Manager
L6	**P17**	**0**	**Collaboration Management**
L6	P17	1	Project Manager
L6	P17	2	Issue Manager
L6	P17	3	Configuration Manager
L6	P17	4	Wiki Manager
L6	P17	5	Cloud collaboration Manager
L6	P17	6	Creativity Support Manager
L6	P17	7	Conference Manager
L6	P17	8	Calendar Manager
L6	P17	9	Message Manager
L6	P17	10	Survey Manager
L6	P17	12	Social networking Manager
L6	**P18**	**0**	**User Support**
L6	P18	1	Product Support Manager
L6	P18	2	Training Manager

L7 Knowledge Layer consists of components (typically reusable) for management of the work at business applications knowledge management level. The components here are split in five packages: P19 General Intelligence, P20 Perception, P21 Natural language processing, P22 Social Intelligence, and P23 Reasoning.

TABLE IX. LAYER 7

Layer	Pack	Comp	Layer/ Package/ Component
L7	**0**	**0**	**Knowledge Layer**
L7	**P19**	**0**	**General Intelligence**

Layer	Pack	Comp	Layer/ Package/ Component
L7	P19	1	Intelligent Agents Manager
L7	P19	2	Computer-Aided Design
L7	P19	3	5GL Manager
L7	P19	4	Intelligent Robots
L7	P19	5	Intelligent Computers
L7	P19	6	Neural Networks
L7	**P20**	**0**	**Perception**
L7	P20	1	Speech recognition Manager
L7	P20	2	Facial recognition Manager
L7	P20	3	Object recognition Manager
L7	**P21**	**0**	**Natural language processing**
L7	P21	1	Information Retrieval Manager
L7	P21	2	Machine Translation Manager
L7	P21	3	Question Answering Manager
L7	P21	4	Text Mining manager
L7	**P22**	**0**	**Social Intelligence**
L7	P22	1	Game Manager
L7	P22	2	Sentiment Analysis Manager
L7	P22	3	Virtual Reality Manager
L7	**P23**	**0**	**Reasoning**
L7	P23	1	Expert Systems
L7	P23	2	Case-Based Reasoning Manager
L7	P23	3	Fuzzy Logic Manager
L7	P23	4	Probability Manager
L7	P23	5	Machine Learning Manager

L8 Management & Planning Layer consists of components (typically reusable) for management of the work at the business applications management &planning level. The components here are split in two packages: P19 Middle management, P20 Strategic management.

TABLE X. LAYER 8

Layer	Pack	Comp	Layer/ Package/ Component
L8	**0**	**0**	**Management & Planning Layer**
L8	**P24**	**0**	**Middle management**
L8	P24	1	Sales management
L8	P24	2	Annual budgeting
L8	P24	3	Capital Investment Analysis
L8	P24	4	Relocation analysis
L8	P24	5	Sales region analysis
L8	P24	6	Production scheduling
L8	P24	7	Cost analysis
L8	P24	8	Pricing/ probability analysis
L8	P24	9	Contract cost analysis
L8	**P25**	**0**	**Strategic management**
L8	P25	1	5-year sales trend forecasting
L8	P25	2	5-year operating plan
L8	P25	3	5-year budget forecasting
L8	P25	4	Profit planning
L8	P25	5	Personnel planning

L9 Integration Layer includes integration definitions and products for integrating systems, processes, and services developed within the organization, by partners or by third parties. This layer includes: P26 Enterprise service bus (ESB) and Semantic ESB, which are main communication instruments of CPAP; P27 Presentation management components, which are key I/O instruments of CPAP; P28 Integration Definitions, including the conventions for integration of external and internal users with CPAP components; P29 TIS integrators, including all components for integration of end user applications with the same functionality but with different implementation and architecture.

TABLE XI. LAYER 9

Layer	Pack	Comp	Layer/ Package/ Component
L9	**0**	**0**	**Integration Layer**
L9	**P26**	**0**	**ESB**
L9	P26	1	Semantic ESB
L9	P26	2	ESB
L9	**P27**	**0**	**Presentation Management**
L9	P27	1	KBASE GUID
L9	P27	2	Developer GUID
L9	P27	3	Portal
L9	P27	4	SysAdmin GUID
L9	P27	5	End User GUID
L9	P27	6	GIS GUID
L9	P27	7	Reports GUID
L9	**P28**	**0**	**Integration Definitions**
L9	P28	1	Information Systems Integration Definitions
L9	P28	2	KBASE Integration Definitions
L9	P28	3	Template IS Integration Definitions
L9	P28	4	Embedded Systems Integration Definitions
L9	**P29**	**0**	**TIS Integrators**
L9	P29	1	Project Integrator
L9	P29	2	Authorization Integrator
L9	P29	3	Product Support Integrator
L9	P29	4	Assets Integrator
L9	P29	5	Financial Integrator
L9	P29	6	Payment Integrator
L9	P29	7	Register Integrator
L9	P29	8	Document Integrator
L9	P29	9	HR Integrator
L9	P29	10	CRM Integrator
L9	P29	11	ERP Integrator
L9	P29	12	Supply chain Integrator

As result of CPAP introduction in practice two important changes for the software companies are identified: (1) the reusability of components is radically improved due to the realization of full design before implementation and the clear separation between standardized and predefined components of the domain independent, domain dependent and problem solving phases; (2) improvement of documentation and reduction of the gap between design and development by introducing full design models descriptions, and highly automated code generation based on the design models.

VIII. RESULTS

The three key results achieved in the related works confirm that the proposed research is successful, i.e. **Result 1** the GUI management component in BeC.2 was realized by IL Java for 28 man-months (2 business analysts and 12 GUI programmers). The same product in BeC.3 was developed by intelligent self-organizing tool for 14 man-months (involving 4 business analysts and 3 GUI programmers); **Result 2** Bulgarian Morphological Processor was realized in IL Pascal with 12,000 source lines of code (including IT experts - 120 man-days, and Linguists - 40 man-days). The same Morphological Processor was realized in 5GL Net with 800 source lines of code (including IT experts - 16 man-days, and Linguists - 44 man-days); **Result 3**. During the design of the prototype OSCAR realized following the Method for automated programming of robots, 61,9% of the efforts were dedicated to the domain independent phase (performed once in the lifecycle), 37,26 % - to the domain dependent phase

(performed once for each new domain area), and only 0,84% - to the problem solving phase (performed daily during the lifetime of the product).

The **KBASE method** and related SL and CPAP represent an original contribution to the SE area due to the following main reasons: (1) generation of machine code from business architectures, (2) replacement of the IT experts with DA experts in key roles of SDP, and (3) new unified process. The **SL** (as a combination of GML, 5GL, and NL specification techniques) and **CPAP** (as a combination of knowledge based automation and cloud computing), are new ideas that bring together different strands of programming language theory, and SE.

IX. Conclusion

The two ***economic challenges*** identified here (**inefficient SDP for EIS**, and **increasing demand of programmers**) have the potential to be solved to a great extent if the paradigm Programming without programmers, and the proposed method are successfully realized. KBASE success could generate a number of ***possible effects***, i.a.: (1) replacement of widely disseminated industrial tools for IS programming with new generation tools; (2) improvement of SDP efficiency (in accordance with the Related works results), including reduction of development time, and decrease of project expenses; (3) decrease of programmers and increase of business analysts, and DA experts in the SDP; (4) modifications required in the BSc and MSc Programs in IT; and (5) the modifications required in international IT standards.

References

[1] A.Ralston, E.D. Reilly, D. Hemmendinger, Encyclopedia of Computer Science, 4th edition, John Wiley and Sons Ltd., 2003.

[2] Axelos Limited, Managing Successful Projects with PRINCE2. The Stationery Office, 2017

[3] EU INCO Copernicus 96/0231. Intelligent Product Manuals (IPM).

[4] EU OPAC. K10-31-1 / 07.09.2010. Program: Extension of Bulgarian Governmental administrative eServices Development (BeG).

[5] F.Liu et. all., NIST Cloud Computing Reference Architecture, Gaithersburg: National Institute of Standards and Technology Special Publication 500-292, US Department of Commerce, 2011

[6] I.Stanev, "Method for Automated Programming of Robots," Knowledge Based Automated Software Engineering, Cambridge Scholars Press, Cambridge, pp.67 – 85, 2012.

[7] I.Stanev, K.Grigorova, "KBASE Unified Process," Knowledge Based Automated Software Engineering, Cambridge Scholars Publishing, Cambridge, pp. 1 – 19, 2012

[8] I.Stanev, M. Koleva, Architecture Knowledge for Software Generation, International Journal of Education and Information Technologies, ISSN: 2074-1316 Volume 12, 2018 Pp. 46-57.

[9] I.Stanev, M. Koleva, Bulgarian eGovernment Information System Based on the Common Platform for Automated Programming – Technical Solution. Proceedings of Tenth International Conference ISGT'2016, Sofia, Bulgaria, Sep 30-Oct 1, 2016, ISSN: 1314-4855. Pp.125 - 134

[10] I.Stanev, M. Koleva, Bulgarian Health Information System based on the Common Platform for Automated Programming, Tenth Mediterranean Conference on Information Systems (MCIS), Paphos, Cyprus, September 2016. Pp. 71.1 – 71.7.

[11] I.Stanev, M. Koleva, KBASE Technological framework - Requirements, In: Proceedings of ICSII 2015 17th International Conference on Semantic Interoperability and Integration, Rome, 2015. . Pp. 634 - 637

[12] I.Stanev, M. Koleva, Knowledge Based Automated Software Engineering Platform Used for the Development of Bulgarian E-Customs. In: Proceedings of 19th International Conference on Applied Computer Science and Engineering (ICACSE 2017). London, UK, Dec 18-19, 2017, 19 (12) Part XVII. Pp. 1900 - 1906

[13] I.Stanev. Method for automated programming of Robots. PhD Thesis. Department Informatics and Information Technologies. University of Ruse. Ruse. Bulgaria. 2014. Pp.141.

[14] J.-D. Lovelock et all., "Forecast alert: IT spending, worldwide", 1Q18 update, https://www.gartner.com/doc/3870395, April 2018.

[15] K.C. Laudon, J.P. Laudon, Management Information Systems: Managing the Digital Firm 15th Edition, Pearson, 2017.

[16] K.Holtzblatt and H.Beyer, Contextual Design, Second Edition: Design for Life. Morgan Kaufmann Publishers. San Francisco, 2016.

[17] M.B.Prescott, S.A.Conger, (1995). Information technology innovations: a classification by IT locus of impact and research approach. ACM SIGMIS Database, 26(2-3), 20-41.

[18] Object Management Group (OMG), Software & Systems Process Engineering Metamodel Specification (SPEM) v.2.0. http://www.omg.org/spec/SPEM/2.0 /, 2008

[19] OMG, Business Process Model and Notation v.2.0.2. http://www.omg.org/spec/BPMN , 2014

[20] OMG, Case Management Model and Notation v1.1. http://www.omg.org/spec/CMMN , 2016

[21] OMG, Decision Model and Notation v1.1. http://www.omg.org/spec/DMN , 2016

[22] OMG, Unified Modeling Language v.2.5.1. http://www.omg.org/spec/UML/ , 2017

[23] P.Abrahamson, O.Salo, J.Ronkainen, J.Warsta, Agile software development methods: Review and analysis (Technical report). VTT. 478., 2002

[24] P.Kruchten, The Rational Unified Process: an Introduction, Addison-Wesley, 2000.

[25] P.Mell and T.Grance, "The NIST Definition of Cloud Computing," Gaithersburg: National Institute of Standards and Technology NIST Special Publication 800-145 US Department of Commerce. p.7, 2011

[26] Project EU IST-1999-20162 Development and Applications of New Built-in-Test Software Components in European Industries, Software Architecture, 2003

[27] Project Management Institute, A Guide to the Project Management Body of Knowledge Sixth Edition, 2017.

[28] Project ДФНИ-И02/13 Development of method and Tool for Generation, Verification and Behavior Evaluation of Business Processes from Selected Domain Area (BPGen), 2018

[29] R.Bouzidi, F.Nader, R.Chalal, Towards a classification of information technologies, In: Proceedings of the International Conference on Computing for Engineering and Sciences (ICCES '17), July 22–24, 2017, Istanbul, Turkey, ACM New York, pp 24-28.

[30] S.Nevo, D.Nevo, P.Ein-Dor, P. (2009), "Thirty Years of IS Research: Core Artifacts and Academic Identity," Communications of the Association for Information Systems: Vol. 25 , Article 24.

[31] U.S. Bureau of Labor Statistics, Occupational Outlook Handbook, https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm#tab-6.

[32] W3C, Web Ontology Language v.2, https://www.w3.org/standards/techs/owl#w3c_all , 2012.